The SuperH-3, part 6: Division

devblogs.microsoft.com/oldnewthing/20190812-00

August 12, 2019

Raymond Chen

The SH-3 does not have a simple "divide two integers please" instruction. Rather, it has a collection of instructions that let you build a division operation yourself.

```
DIV0U               ; prepare for unsigned division
DIV0S   Rm, Rn      ; prepare for signed division Rn ÷ Rm
DIV1    Rm, Rn      ; generate 1 bit of the quotient Rn ÷ Rm
```

To begin an integer division operation, you execute either a `DIV0U` or `DIV0S` instruction, depending on whether you want a signed or unsigned division.

You then perform a number of `DIV1` instructions equal to the number of bits of quotient you need, mixed in with other instructions are outlined in the programmer's manual. After running the desired number of iterations, the result is in either the *Rn* register or in the register you accumulated the results into, depending on the specific algorithm you used.

These instructions do not attempt to handle division by zero or division overflow. They will simply generate nonsense results. If preventing division by zero or overflow is important to you, you will have to check for them yourself explicitly.

I'm not going to go into the fine details of how these instructions operate. They use *Rm* and *Rn* to record the state of the division, with three additional bits of state recorded in the *M*, *Q*, and *T* flags.

It's basically magic.

In practice, you won't see the compiler generate these instructions anyway. Instead, the compiler is going to do one of the following:

- If dividing by a constant power of 2, use a shift instruction.
- If dividing by a small constant, multiply by $2^{32} \div n$ and extract the high 32 bits of the result.
- Otherwise, call a helper function in the runtime library.

Phew, that was crazy.

Next time, we'll return to the relative sanity of bitwise logical operations.

Bonus chatter: If you want to see how one compiler implements division on the SH-3 (and if you are okay with being exposed to GPL source code), you can take a look at how GCC implements division in its runtime library.

Raymond Chen

Follow